UNITED STATES PATENT AND TRADEMARK OFFICE

BEFORE THE BOARD OF PATENT APPEALS
AND INTERFERENCES

Ex parte Dennis Steven DeLorme, Margaret Rose Fenion, Robert Thaddeus Gintowt, Alan Leon Levering, Thomas Marcus McBride, Michael John Oberholtzer, Jeffrey John Parker, Richard Michael Theis, and Jerry Leroy Von Berge

---

Appeal No. _____
Application No. 10/777,869

APPEAL BRIEF

Attorney Docket No. ROC920040004US1 PATENT
Confirmation No. 6102

CERTIFICATE OF ELECTRONIC TRANSMISSION
I hereby certify that this correspondence for Application No. 10/777,869 is being electronically transmitted to Technology Center 2161 via EFS-WEB, on April 1, 2008.

/Scott A. Stinebruner/ April 1, 2008
Scott A. Stinebruner, Reg. No. 38,323 Date

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

Applicant:	Dennis Steven DeLorme et al.	Art Unit: 2161
Application No.:	10/777,869	Examiner: Paul Kim
Filed:	February 12, 2004	
For:	METHOD OF CONVERTING A FILESYSTEM WHILE THE FILESYSTEM REMAINS IN AN ACTIVE STATE	

Mail Stop Appeal Brief - Patents
Commissioner for Patent
P.O. Box 1450
Alexandria, VA 22213-1450

APPEAL BRIEF

I. REAL PARTY IN INTEREST

This application is assigned to International Business Machines Corporation, of Armonk, New York.

II. RELATED APPEALS AND INTERFERENCES

There are no related appeals or interferences.

III. STATUS OF CLAIMS

Claims 1-35 are pending in the Application, stand rejected, and are now on appeal.

IV. STATUS OF AMENDMENTS

There have been no amendments filed subsequent to the final rejection mailed November 1, 2007.

V. SUMMARY OF CLAIMED SUBJECT MATTER

Applicant's invention is generally directed to a filesystem conversion process that does not require shutting down a filesystem to perform the conversion, ensures all objects within the filesystem are converted, and, from the perspective of a user, does not impact the performance and operation of the file system (Application, page 5, lines 2-5).

Filesystem conversion processes conventionally have been used to update filesystems from one type to another, e.g., to upgrade filesystems to newer, better performing, and more feature rich versions. Conventional conversion processes have often required filesystems to be shut down and restarted in order to complete a filesystem conversion. Doing so, however, makes the filesystem unavailable for productive use, and in many applications, such as high availability and mission critical applications, shutting down a filesystem for any amount of time is highly undesirable. (Application, page 2, line 17 to page 3, line 13).

Embodiments consistent with the invention, on the other hand, address the drawbacks of conventional filesystem conversion processes by enabling a filesystem to be converted while maintained in an active state. Independent claims 1 and 24, for example, respectively recite a method (Application, page 13, lines 12-14) and a program product (Application, page 9, line 16 to page 10, line 11) for converting a filesystem from a first type to a second type. (Application, page 5, lines 6-11). To perform the conversion, a list is created of directories of the first type in the filesystem to convert (Application, page 16, lines 3-4, page 16, line 10 to page 17, line 8, Fig. 4, block 402, Fig. 5, blocks 502-510). Then, each directory in the list is converted to the second type while maintaining the filesystem in an active state (Application, page 16, lines 1-9, page 17, line 21 to page 21, line 15, Fig. 3, blocks 306-308, Fig. 7, blocks 702-734).

Independent claims 20, 25 and 26 respectively recite a method (Application, page 13, lines 12-14), program product (Application, page 9, line 16 to page 10, line 11) and apparatus (Application, page 7, lines 9-19, Fig. 1, block 10) for converting a filesystem from a first type to a second type. (Application, page 5, lines 6-11). To perform the conversion, a conversion process is executed to convert each directory of the first type in the filesystem into the second type while maintaining the file system in an active state. (Application, page 16, lines 1-9, page 17, line 21 to page 21, line 15, Fig. 3, blocks 306-308, Fig. 7, blocks 702-734) Then, the

conversion process is terminated when every directory of the first type in the filesystem has been converted to the second type. (Application, page 21, lines 13-15, Fig. 7, block 734).

For program product claims 24 and 25, support for the recited program code and signal bearing medium can be found at page 9, line 16 to page 10, line 11 of the Application. For apparatus claim 26, support for the claimed processor and memory can be found at page 7, lines 20-21 and Fig. 1, blocks 12 and 14 of the Application.

Other support for the claimed subject matter may generally be found in Figs. 3-5 and 7-8 and the accompanying text at pages 13-24 of the Application as filed. In addition, it should be noted that, as none of the claims recite any means plus function or step plus function elements, no identification of such elements is required pursuant to 37 CFR §41.37(c)(1)(v). Furthermore, there is no requirement in 37 CFR §41.37(c)(1)(v) to provide support for the subject matter in the separately argued dependent claims, and so no discussion of any of these claims is provided.

VI. GROUNDS OF REJECTION TO BE REVIEWED ON APPEAL

A. Claims 1-2, 8, 20-22 and 24-26[1] stand rejected under 35 U.S.C. § 103(a) as being unpatentable over U.S. Patent Publication No. 2003/0037302 to Dzienis (Dzienis) in view of U.S. Patent No. 5,625,804 to Cooper (*Cooper*).

B. Claims 3-5, 9, 23, 28 and 34 stand rejected under 35 U.S.C. § 103(a) as being unpatentable over D*zienis* in view of *Cooper* and further in view of Official Notice.

C. Claim 6 stands rejected under 35 U.S.C. § 103(a) as being unpatentable over *Dzienis* in view of *Cooper* and further in view of U.S. Patent No. 5,873,097 to Harris et al. (*Harris*).

D. Claim 7 stands rejected under 35 U.S.C. § 103(a) as being unpatentable over *Dzienis* in view of *Cooper* and further in view of U.S. Patent No. 6,571,231 to Sedlar (*Sedlar*).

E. Claims 10-11, 18-19 and 30 stand rejected under 35. U.S.C. § 103(a) as being unpatentable over *Dzienis* in view of *Cooper* and further in view of West et al. (NPL, "Batch Processing" excerpt from "Sams Teach Yourself Macromedia Fireworks MX in 24 Hours"), published on 4 December 2002 (*West*), and *Harris*.

[1] The Examiner's first rejection, made at paragraph 5 of the Final Office Action, lists claims 1-2, 7-9, 20-22, 24-26 and 28; however, claims 9 and 28 were addressed along with the rejection of claims 3-5, 23 and 34 at paragraph 13 of the Final Office Action, and claim 7 was rejected in paragraph 17 based upon an additional reference to *Sedlar,* so Applicant assumes that the Examiner's recitation of the rejections was in error.

F. Claims 14 and 32 stand rejected under 35. U.S.C. § 103(a) as being unpatentable over *Dzienis* in view of *Cooper* and further in view of U.S. Patent Application Publication No. 2003/0217057 by Kuroiwa et al. (*Kuroiwa*).

G. Claims 12, 15-17, 29 and 31 stand rejected under 35. U.S.C. § 103(a) as being unpatentable over *Dzienis* in view of *Cooper* and further in view of U.S. Patent No. 6,728,907 to Wang et al. (*Wang*).

H. Claim 13 stands rejected under 35. U.S.C. § 103(a) as being unpatentable over *Dzienis* in view of *Cooper* and *Wang*, and further in view of *Official Notice*.

I. Claim 27 stands rejected under 35. U.S.C. § 103(a) as being unpatentable over *Dzienis* in view of *Cooper* and further in view of Dubinski (NPL, "Non-recursive tree walks"), by John Dubinski, published on 1 May 1996 (*Dubinski*).

J. Claims 33 and 35 stand rejected under 35. U.S.C. § 103(a) as being unpatentable over *Dzienis* in view of *Cooper* and further in view of U.S. Patent No. 6,338,072 to Durand et al. (*Durand*).

VII. ARGUMENT

Applicant respectfully submits that the Examiner's rejections of claims 1-35 are not supported on the record, and should be reversed. All such claims have been rejected as being obvious over the prior art cited by the Examiner. A *prima facie* showing of obviousness, however, requires that the Examiner establish that the differences between a claimed invention and the prior art "are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art." 35 U.S.C. §103(a). Such a showing requires that all claimed features be disclosed or suggested by the prior art. Four factors generally control an obviousness inquiry: 1) "the scope and content of the prior art"; 2) the "differences between the prior art and the claims"; 3) "the level of ordinary skill in the pertinent art"; and 4) objective evidence of non-obviousness. KSR International Co. v. Teleflex Inc., 127 S. Ct. 1727, 1734 (2007) (quoting Graham v. John Deere Co. of Kansas City, 383 U.S. 1, 17-18 (1966)). Furthermore, while there is no rigid requirement for an explicit "teaching, suggestion or motivation" to combine references, there must be some evidence of "a reason that would have prompted a person of ordinary skill in the relevant field to combine the elements in the way the claimed new invention does" in an obviousness determination. KSR, 127 S. Ct. at 1731.

Applicant respectfully submits that, in the instant case, the Examiner has failed to establish a *prima facie* case of obviousness as to claims 1-35, as such, the rejections thereof should be reversed. Applicant's remarks in rebuttal to the Examiner's rejections are presented below, starting with the relevant independent claims and followed up by a discussion of selected dependent claims. In some cases, specific discussions of particular claims are not made in the interests of streamlining the appeal. The omission of a discussion with respect to any particular claim, however, should not be interpreted as an acquiescence as to the merits of the Examiner's rejection of the claim, particularly with respect to claims reciting features that are addressed in connection with the rejections applied to other claims pending in the appeal.

A. Claims 1-2, 8, 20-22, and 24-26 are non-obvious over *Dzienis* and *Cooper*.

As an initial procedural matter, Applicant notes that the Examiner previously rejected all of the claims based primarily on the combination of *Sedlar* and *Cooper* (rejections that Applicant previously appealed), and that in lieu of maintaining the appeal, the Examiner chose to reopen prosecution, issuing new rejections that essentially replaced *Sedlar* with *Dzienis* as the primary reference. As will be discussed in greater detail below, however, *Dzienis* appears to be even less relevant to the claims at issue than *Sedlar*, and as such, the rejections of all claims should be reversed, and the case passed to issue.

Independent Claim 1

Claim 1 generally recites a method for converting a filesystem from a first type to a second type. The method comprises the steps of: generating a list of directories of the first type in the filesystem to convert, and converting each directory in the list to the second type while maintaining the file system in an active state.

In rejecting claim 1, the Examiner has replaced the former rejection based on the combination of *Sedlar* and *Cooper* with a rejection based on the combination of *Dzienis* and *Cooper*. *Dzienis*, however, is not even directed to the conversion of a filesystem from a first type to a second type, or even to the conversion of directories in a filesystem. Instead, the reference discloses nothing more than a method of converting files in a filesystem to a common file format. Paragraph [0003] of *Dzienis* states that the disclosed invention is directed to "automatically converting a plurality of document files in various native formats to a single

common format." The preferred embodiment, for example, is used in a document management system to convert word processing, spreadsheet, and other files into an imaging format such as TIFF (*Dzienis*, paragraph [0031]).

To the extent that directories are disclosed in *Dzienis*, the directories serve as little more than containers for the files being converted to the common format. Paragraphs [0042]-[0045] disclose that a user selects a target directory for conversion, and the system scans that directory as well as any sub-directories in the target directory, and calculates the number of files, as well as generates and displays a list of the files to be converted (*Dzienis*, paragraph [0044]). Paragraph [0047] also discloses that during TIFF conversion, each file to be converted is first copied from the target directory to a local, temporary directory. In all instances, however, the directory itself is never converted.

Turning specifically to the Examiner's rejection of claim 1, the Examiner first asserts that *Dzienis* discloses the generation of a list of directories to convert, specifically at paragraph [0040]. This paragraph states in its entirety:

> *[0040] For processing, initially an inventory is performed by scanning of the directory containing files to be converted and calculating the number and types of different files. This provides the user with complete statistics about the data to be translated into a common file format.*

This passage, however, does not disclose or suggest generating any list of directories, much less a list of directories to convert. Instead, the passage is merely directed to generating a list of files to be converted, where those files happen to be in a particular directory. As such, *Dzienis* does not disclose or suggest "generating a list of directories of the first type in the filesystem to convert," as alleged by the Examiner. The Examiner has not, and could not, rely on *Cooper* for disclosing this concept, so the Examiner has failed to establish that the combination of *Dzienis* and *Cooper* discloses or suggests each and every feature of claim 1. No *prima facie* case of obviousness has therefore been established as to claim 1.

The Examiner then asserts that *Dzienis* discloses converting every directory in a list to a second type, citing paragraph [0042]. This paragraph reads in its entirety:

> *[0042] If there are any discrepancies, complete information about the data is displayed so that the user can identify the errors and take the appropriate corrective action. The file conversion is then performed on each file for every file that is supported. In order to accomplish conversion, each file is opened, processed and submitted to the print driver for conversion. A final integrity check of the data is made and the user receives a complete error log.*

This passage, however, merely addresses the conversion of individual files. There is no disclosure or suggestion in the passage, or elsewhere in *Dzienis*, of any conversion of a directory from one type to another, or even to the conversion of a filesystem itself. As such, *Dzienis* does not disclose or suggest "converting each directory in the list to the second type," as alleged by the Examiner. In addition, the Examiner has not, and could not, rely on *Cooper* for disclosing this concept, so the Examiner has failed to establish that the combination of *Dzienis* and *Cooper* discloses or suggests each and every feature of claim 1. No *prima facie* case of obviousness has therefore been established as to claim 1.

The Examiner then implicitly acknowledges that *Dzienis* does not disclose performing a conversion while maintaining a filesystem in an active state by citing *Cooper*. *Cooper*, however, fails to address the shortcomings of *Dzienis*. The Examiner relies on col. 2, lines 44-59 of *Cooper* for allegedly disclosing converting directories in a list to a second type while maintaining a filesystem in an active state. *Cooper*, however, discloses the conversion of individual records to different formats while maintaining a multiprocessor system in an active state. *Cooper* does not disclose that the records are directories, files, and in fact, does not mention "directories", "files", "folders", or any other analogous structures related to filesystem. Moreover, *Cooper* does not disclose maintaining a filesystem in an active state while changing directories in that filesystem to different types.

In addition, the conversion of records in *Cooper* does not even include the generation of any list of records to convert. Instead, records are converted in *Cooper* as they are accessed by individual processors (*see, e.g.,* Figs. 2 and 4a). *Cooper* does not generate any list of records to be converted, and in fact, does not retrieve records to be converted from a list or otherwise rely

on any type of list in connection with converting records. Furthermore, *Cooper* does not even disclose that the individual records being converted are linked to one another as would be found in a list of directories in a filesystem.

Claim 1 recites converting "each directory in the list," and as such, the fact that *Cooper* discloses converting the format of individual records falls short of disclosing or suggesting the conversion of directories that are identified in a "list of directories . . . to convert," as required by claim 1.

As such, the combination of *Dzienis* and *Cooper* falls short of disclosing each and every feature of claim 1. In particular, neither reference discloses or suggests the generation of a "list of directories . . . to convert", and neither reference discloses or suggests converting directories "in [a] list [to convert]" to a different type.

In addition, neither reference even discloses or suggests the fundamental concept recited in claim 1, that of converting a filesystem to a different type while maintaining that filesystem in an active state. *Dzienis* does not address this particular feature, as the reference is merely directed to converting individual files to a common format, and the reference is not at all concerned with modifying or converting any underlying filesystem. In addition, maintaining a multiprocessor system in an active state while converting individual records to a different format (as disclosed in *Cooper*), does not specifically disclose or suggest the conversion of a filesystem to a different type while that filesystem is in an active state.

Applicant also respectfully submits that the rejection is improperly reliant on hindsight. Specifically, neither reference discloses or suggests converting a filesystem to a different type, much less doing so while maintaining that filesystem in an active state. *Dzienis* merely discloses the conversion of individual files to a common format, and to the extent that a filesystem is disclosed, there is no manipulation or conversion of the filesystem itself. *Cooper* does not even disclose filesystems, directories, or any analogous concepts, and the fact that the reference merely discloses converting individual records to different formats falls short of disclosing or suggesting the conversion of a filesystem. Applicant therefore submits that *Cooper* does not provide any motivation to one of ordinary skill in the art to modify *Dzienis* to implement a runtime filesystem conversion process. Applicant also submits that Examiner has not otherwise provided any credible reason why one of ordinary skill in the art would be motivated to modify *Dzienis* to implement a runtime filesystem conversion process. Absent any evidence of such a

motivation, or even any professed reason to make a modification, the rejection is necessarily reliant on hindsight, and as such, the rejection cannot be maintained.

Applicant therefore respectfully submits that claim 1 is non-obvious over the combination of *Dzienis* and *Cooper*. Reversal of the Examiner's rejection, and allowance of claim 1, and of claims 2-19 that depend therefrom, are therefore respectfully requested.

Independent Claim 20

Claim 20 generally recites a method for converting a filesystem from a first type to a second type. The method comprises the steps of: executing a conversion process to convert each directory of the first type in the filesystem into the second type while maintaining the filesystem in an active state, and terminating the conversion process when every directory of the first type in the filesystem has been converted to the second type.

In rejecting claim 20, the Examiner again relies on *Dzienis* and *Cooper*. However, as discussed above in connection with claim 1, the combination of *Dzienis* and *Cooper* does not disclose or suggest a method of converting a filesystem from a first type to a second type while maintaining the filesystem in an active state. *Dzienis* discloses only the conversion of individual files (which may be stored in the directories of a filesystem) to a common format, but does not disclose or suggest converting any filesystem to another type, much less doing so while maintaining that filesystem in an active state. Likewise, *Cooper*, while disclosing converting data records to different formats while maintaining a multiprocessor system in an active state, does not even mention filesystems, and certainly does not suggest that the data record conversion process disclosed therein can be used to convert a filesystem.

Applicant also respectfully submits that the rejection of claim 20 is improperly reliant on hindsight. In order to establish a *prima facie* case of obviousness for claim 20, the Examiner must present some objective evidence of a motivation in the art to combine *Dzienis* and *Cooper* to convert a filesystem from one type to another while maintaining that filesystem in an active state. *Dzienis* itself does not provide any such motivation, given that there is no discussion in the reference directed to converting a filesystem to a different type. *Cooper*, likewise, fails to provide any such motivation, given that the reference does not discuss filesystems, or otherwise suggest that the data record conversion process disclosed in the reference could be used in converting a filesystem. Neither reference even appreciates the desirability of converting a

filesystem while maintaining that filesystem in an active state. In addition, the Examiner has cited no other reference providing any of the motivation lacking in *Dzienis* and *Cooper*. Applicant submits that the Examiner has failed to provide any objective reason why one of ordinary skill in the art would be motivated to combine *Dzienis* and *Cooper* to convert a filesystem while maintaining that filesystem in an active state.

Applicant submits that in this case, Applicant's disclosure has simply been used as a blueprint, given that *Dzienis* and *Cooper* otherwise have no relationship to one another in terms of the problems addressed thereby or the solutions used to address such problems. Neither reference even addresses the same problem as that faced by Applicant, and Applicant therefore respectfully submits that no *prima facie* case of obviousness has been established for claim 20. Reversal of the Examiner's rejection and allowance of claim 20, and of claims 21-23 which depend therefrom, are therefore respectfully requested.

Independent Claim 24

Claim 24 recites in part program code configured to generate a list of directories of a first type in a filesystem to convert, and convert each directory in the list to a second type while maintaining the filesystem in an active state. As discussed above in connection with claim 1, this combination of features is not disclosed or suggested by the combination of *Dzienis* and *Cooper*. Accordingly, claim 24 is non-obvious over these references for the same reasons as presented above for claim 1. Reversal of the Examiner's rejections, and allowance of claim 24 are therefore respectfully requested.

Independent Claims 25 and 26

Claims 25 and 26 recite in part program code configured to initiate a conversion process to convert each directory of a first type in a filesystem into a second type while maintaining the filesystem in an active state, and terminate the conversion process when every directory of the first type in the filesystem has been converted to the second type. As discussed above in connection with claim 20, this combination of features is not disclosed or suggested by the combination of *Dzienis* and *Cooper*. Accordingly, claims 25 and 26 are non-obvious over these references for the same reasons as presented above for claim 20. Reversal of the Examiner's

rejection and allowance of claims 25-26, and of claims 27-35 which depend therefrom, are therefore respectfully requested.

Dependent Claims 2, 8 and 21

Claims 2, 8 and 21 are not argued separately.

Dependent Claim 22

Claim 22 depends from claim 20, and additionally recites the step of generating a list of directories of the first type in the filesystem to convert. In rejecting claim 22, the Examiner groups this claim with claim 1. Applicant assumes that the Examiner again relies on *Dzienis*, paragraph [0040], for allegedly disclosing generating a list of directories. However, as Applicant discussed above in connection with claim 1, *Dzienis* does not disclose or suggest generating a list of "directories . . . to convert." The cited passage addresses at the most the generation of a list of files to be converted to a common format, and *Dzienis* otherwise does not convert or modify a directory or any other component of a filesystem. *Dzienis*, having no appreciation for filesystem conversion, therefore does not disclose or suggest any such type of list. In fact, given that the claimed list contains those directories "to convert" during a filesystem conversion process, that list will change as directories are removed after they are successfully converted. Neither *Dzienis* nor *Cooper* discloses or suggests any analogous functionality for maintaining a list of directories still awaiting conversion. Accordingly, no *prima facie* case of obviousness has been established for claim 22, and claim 22 is therefore non-obvious over the references cited by the Examiner. Reversal of the Examiner's rejection of claim 22 is therefore respectfully requested.

B. Claims 3-5, 9, 23, 28 and 34 are non-obvious over *Dzienis*, *Cooper* and *Official Notice*.

Dependent Claims 3 and 23

Claim 3 depends from claim 1 additionally recites that the step of converting further includes the steps of: retrieving an identifier of a directory in the list, converting the directory to a second-type directory, and activating the second-type directory. Claim 23 depends from claim 20 and recites similar subject matter. In rejecting these claims, the Examiner adds *Official Notice* to *Dzienis* and *Cooper*. The Examiner simply disregards the fact that neither reference

discloses activating a directory with the conclusory allegation that it would have been obvious to activate a directory. Such a conclusory observation, however, absent any supporting evidence, falls far short of raising to the level necessary to sustain a *prima facie* case of obviousness. Reversal of the Examiner's rejections of claims 3 and 23 is therefore respectfully requested.

Dependent Claims 4-5, 9 and 28

Claims 4-5, 9 and 28 are not argued separately.

Dependent Claim 34

Claim 34 depends from claim 26 and additionally recites that the program code is further configured to, for a particular directory already converted, convert the particular directory back to the first type. In rejecting this claim, the Examiner adds *Official Notice* to *Dzienis* and *Cooper*, arguing that it would have been obvious to revert a directory already converted back to its original type. Once again, the Examiner's reliance on *Official Notice* has no support on the record, and is simply a conclusory argument used only because *Dzienis* and *Cooper* are admittedly silent on the concept. *Cooper*, in particular, does not disclose any conversion back to an original format. Applicant submits that, in the least, the Examiner is required to provide some evidence that one of ordinary skill in the art would in fact be motivated to convert a directory back to its original type after it has already been converted. The Examiner cannot meet this burden in this instance, and as such, the Examiner has failed to meet the burden required to establish a *prima facie* case of obviousness. Reversal of the Examiner's rejection of claim 34 is therefore respectfully requested.

C. Claim 6 is non-obvious over *Dzienis*, *Cooper* and *Harris*.

Claim 6 depends from claim 1 and therefore incorporates all of the subject matter thereof. *Harris* does not and has not been asserted by the Examiner to disclose or suggest the underlying features recited in claim 1. Accordingly, irrespective of whether or not *Harris* discloses or suggests any of the additional features of claim 6, claim 6 is patentable over the cited references for the same reasons as claim 1. Reversal of the Examiner's rejection, and allowance of claim 6, are therefore respectfully requested.

D. Claim 7 is non-obvious over *Dzienis*, *Cooper* and *Sedlar*.

Claim 7 depends from claim 1, and additionally recites that the step of generating further includes the steps of:

a) adding a root directory as a current entry in the list;

b) identifying a child directory of the current entry in the list;

c) appending the identified child directory to the list;

d) repeating steps b) and c) for each child directory within the current entry;

e) changing a next directory in the list immediately following the current entry to be the current entry, if the next directory exists in the list; and

f) repeating steps b) - e) until no next directory exists in the list.

In rejecting claim 7, the Examiner argues that *Sedlar* discloses each of these concepts at Figs. 1-3 and Col. 3, lines 50-61. The cited passages, however, merely disclose a directory links table including a list of directories and the links therebetween. There is nothing in the cited passages that describes or suggests any of the specific steps recited in claim 7, e.g., "adding a root directory," "identifying a child directory," "appending the child directory to the list," and "changing a next directory in the list." The Examiner's analysis of claim 7 is superficial at best, and appears to rely only on the fact that elements such as entries, lists, and directories are disclosed in the cited passages. There is no disclosure of how these elements are processed in the precise manner recited in claim 7. As such, the Examiner has failed to meet the burden required to establish a *prima facie* case of obviousness as to claim 7. Reversal of the Examiner's rejection of claim 7 is therefore respectfully requested.

E. Claims 10-11, 18-19 and 30 are non-obvious over *Dzienis*, *Cooper, West and Harris*.

Dependent Claim 10

Claim 10 depends from claim 1 and additionally recites that the step of converting further includes the steps of: a) creating a second-type root directory, b) creating a second-type directory corresponding to a particular directory in the list, c) generating a respective link in the second-type directory for each child object of the particular directory, d) activating the second-type directory, and e) removing the particular directory from the list.

In rejecting the claim, the Examiner adds *West* to the rejection, arguing that the reference discloses creating the second-type root directory, creating the second-type directory corresponding to the particular directory in the list, and generating a link in the second-type directory for each child object of the particular directory. In making this argument, the Examiner relies on Figs. 18.3 and 18.9 and pp. 1 and 6-7 of *West*. The cited passages, however, merely disclose the creation of directories for entirely different purposes, namely in connection with batch processing of images. The citation of *West* at all is questionable, much less the citation of the reference for the particular claim limitations at issue with claim 10. The Examiner has failed to provide any analysis as to how image processing techniques are relevant to the claimed filesystem conversion process, and in particular, how these techniques are related to the aforementioned steps, which are performed specifically in connection with "converting each directory in [a] list to [a] second type while maintaining [a] filesystem in an active state" (since these steps are recited as being part of the "converting step").

Furthermore, the Examiner relies on *Harris* for allegedly disclosing the activation of a directory and the removal of a directory from the list. Once again, however, the Examiner is improperly relying on hindsight, as the Examiner is merely taking isolated passages from the reference without a consideration of whether there is any reason or motivation for one of ordinary skill in the art to do incorporate those isolated teachings into the proposed combination. The Examiner has not and cannot show any reason that one of ordinary skill in the art would look to *West* and *Harris* to convert a directory to another format using the specific steps recited in claim 10. The Examiner is using Applicant's specification as a blueprint with which to construct the rejection, without any consideration whatsoever into what the references teach as a whole. For this reason, the Examiner's rejection is in error and should be reversed.

Dependent Claim 11

Claim 11 depends from claim 10, and additionally recites the step of creating a data structure associated with the second-type directory, the data structure including a first anchor point that is associated with a parent directory of the directory and a second anchor point associated with a parent directory of the second-type directory.

In rejecting claim 11, the Examiner simply makes a conclusory statement that it would have been obvious to combine the cited references. Under no circumstance should the

Examiner's stated basis for this rejection ever be sufficient to meet the burden required to establish a *prima facie* case of obviousness. The cited passage at col. 5, lines 19-49 of *Harris* is completely irrelevant to the concept of providing two anchor points in a data structure for a directory, and the Examiner provides no analysis of the reference or any explanation why the cited language in the reference applies to the language in claim 11. Reversal of the Examiner's rejection of claim 11 is therefore respectfully requested.

Dependent Claim 18

Claim 18 depends from claim 1 and additionally recites that the step of activating further includes the steps of identifying a data structure associated with the directory, changing the data structure to be associated with the second-type directory, and removing the directory. The Examiner cites *Harris*, col. 5, lines 19-49, but as with claim 11, the passage is of only tangential interest with respect to the claim language, and the Examiner has provided no argument beyond a simple conclusory statement. Applicant submits that the Examiner has not met the burden required to establish a *prima facie* case of obviousness as to claim 18. Reversal of the Examiner's rejection is therefore respectfully requested.

Dependent Claims 19 and 30

Claims 19 and 30 are not argued separately.

F. Claims 14 and 32 are non-obvious over *Dzienis*, *Cooper* and *Kuroiwa*.

Dependent Claim 14

Claim 14 depends from claim 1 and additionally recites determining a usage rate of a particular directory before converting that directory, and postponing converting the particular directory based on the usage rate. Claim 32 depends from claim 26 and recites similar subject matter. In rejecting these claims, the Examiner adds *Kuroiwa* to *Dzienis* and *Cooper*. The cited passages of *Kuroiwa*, however, merely disclose interrupting a conversion process for a content using method based upon CPU load. The passages deal with conversion of media content, and are entirely irrelevant to the concept of determining a usage rate of a directory to be converted when converting a filesystem. The rejection is unfortunately emblematic of the superficial

nature of most of the rejections in the Final Office Action, as there is no evidence presented whatsoever as to how the cited passages would motivate one of ordinary skill in the art to modify *Dzienis* and *Cooper* to arrive at Applicant's claimed invention. The Examiner has again fallen far short of meeting the burden required to establish a *prima facie* case of obviousness as to claims 14 and 32. Reversal of the Examiner's rejections of claims 14 and 32 is therefore respectfully requested.

G. Claims 12, 15-17, 29 and 31 are non-obvious over *Dzienis*, *Cooper* and *Wang*.

Dependent Claims 12 and 29

Claim 12 depends from claim 1 and additionally recites the steps of: detecting a condition for pausing the converting step, and in response to the condition, pausing the converting step. Claim 29 depends from claim 26 and recites similar subject matter. In rejecting this claim, the Examiner adds *Wang* to *Dzienis* and *Cooper*. *Wang*, however, merely discloses a method of diagnosing system crashes, with the cited passages merely disclosing detecting a crash and restarting a system. Once again, the rejections are superficial and conclusory in nature, as there is no discussion of any reason why the cited passages in *Wang* would motivate one of ordinary skill in the art to pause or stop a filesystem conversion process. The Examiner has again fallen far short of meeting the burden required to establish a *prima facie* case of obviousness as to claims 12 and 29. Reversal of the Examiner's rejections is therefore respectfully requested.

Dependent Claims 15 and 31

Claim 15 depends from claim 1 and additionally recites detecting a condition for stopping the converting step, and in response to the condition, stopping the converting step. Claim 31 depends from claim 26 and recites similar subject matter. The Examiner cites the same passages in *Wang* for allegedly teaching these steps as were cited in connection with claims 12 and 29. Also, as with the rejection for claims 12 and 29, the rejections are superficial and conclusory in nature, as there is no discussion of any reason why the cited passages in *Wang* would motivate one of ordinary skill in the art to pause or stop a filesystem conversion process. The Examiner has again fallen far short of meeting the burden required to establish a *prima facie* case of

obviousness as to claims 15 and 31. Reversal of the Examiner's rejections is therefore respectfully requested.

Dependent Claims 16 and 17

Claims 16 and 17 are not argued separately.

H. Claim 13 is non-obvious over *Dzienis, Cooper, Wang* and *Official Notice*.

Claim 13 depends from claim 12, and additionally recites that the condition that triggers the pause is one of a product install on the filesystem; a restore operation involving the filesystem; and a back-up operation involving the filesystem. In rejecting this claim, the Examiner adds *Official Notice* to *Dzienis, Cooper* and *Wang*. The Examiner simply disregards the fact that none of these references discloses the recited conditions for pausing a conversion process with a conclusory allegation that it would have been obvious to pause based upon these conditions. Such a conclusory observation, however, absent any supporting evidence, falls far short of raising to the level necessary to sustain a *prima facie* case of obviousness. In fact, the passages cited in *Wang* in connection with the rejection of claim 12 focus on stopping, rather than pausing, a system, and the conditions that trigger stops are far different than the recited conditions that would trigger a pause in the conversion process recited in claim 13. The Examiner's analysis of claim 13 is once again superficial and conclusory in nature, and insufficient to sustain an obviousness rejection. Reversal of the Examiner's rejection of claim 13 is therefore respectfully requested.

I. Claim 27 is non-obvious over *Dzienis, Cooper* and *Dubinski*.

Claim 27 depends from claim 26 and additionally recites the program code is further configured to non-recursively build a list of directories of the first type. In rejecting this claim, the Examiner adds *Dubinski* to *Dzienis* and *Cooper*, but as with many of the other rejections, the cited passages in *Dubinski* are cited without any supporting analysis (beyond a conclusory statement) suggesting a reason why the teachings of *Dubinski* would motivate one of ordinary skill in the art to arrive at Applicant's claimed invention. The Examiner has not made a *prima facie* case of obviousness, and the rejection of claim 27 should be reversed.

J. Claims 33 and 35 are non-obvious over *Dzienis*, *Cooper* and *Durand*.

Claim 33 depends from claim 32 and additionally recites that the program code is further configured to convert another directory, different than the particular directory, while the particular directory is being used more than the predetermined amount. Claim 35 depends from claim 26, and recites that the program code is further configured to execute at an adjustable priority level. In rejecting these claims, the Examiner adds *Durand* to *Dzienis* and *Cooper*, but as with many of the other rejections, the cited passages in *Durand* are cited without any supporting analysis (beyond a conclusory statement) suggesting a reason why the teachings of *Durand* would motivate one of ordinary skill in the art to arrive at Applicant's claimed invention. The Examiner has not made a *prima facie* case of obviousness, and the rejections of claims 33 and 35 should therefore be reversed.

CONCLUSION

Applicant respectfully requests that the Board reverse the Examiner's rejections of claims 1-35, and that the Application be passed to issue. If there are any questions regarding the foregoing, please contact the undersigned at 513/241-2324. If any other charges or credits are necessary to complete this communication, please apply them to Deposit Account 23-3000.

Respectfully submitted,

April 1, 2008
Date

/Scott A. Stinebruner/
Scott A. Stinebruner
Reg. No. 38,323
WOOD, HERRON & EVANS, L.L.P.
2700 Carew Tower
441 Vine Street
Cincinnati, Ohio 45202
Telephone: (513) 241-2324
Facsimile: (513) 241-6234

VIII. CLAIMS APPENDIX: CLAIMS ON APPEAL (S/N 10/777,869)

Listing of Claims:

1. (Original) A method for converting a filesystem from a first type to a second type, the method comprising the steps of:

generating a list of directories of the first type in the filesystem to convert; and

converting each directory in the list to the second type while maintaining the file system in an active state.

2. (Original) The method of claim 1, further comprising the step of:

sequentially initiating the steps of generating and converting upon initial program load of a computer system utilizing the filesystem.

3. (Original) The method of claim 1, wherein the step of converting further includes the steps of:

retrieving an identifier of a directory in the list;

converting the directory to a second-type directory; and

activating the second-type directory.

4. (Original) The method of claim 3, further comprising the step of:

repeating the steps of claim 3 for each directory in the list.

5. (Original) The method according to claim 1 wherein the list represents a top-down view of the filesystem spanning from a root directory down to an outermost leaf-node.

6. (Original) The method according to claim 5, wherein the step of converting each directory is performed for each directory in an order opposite to that of the list.

7. (Original) The method according to claim 1, wherein the step of generating further includes the steps of:

a) adding a root directory as a current entry in the list;

b) identifying a child directory of the current entry in the list;

c) appending the identified child directory to the list;

d) repeating steps b) and c) for each child directory within the current entry;

e) changing a next directory in the list immediately following the current entry to be the current entry, if the next directory exists in the list; and

f) repeating steps b) - e) until no next directory exists in the list.

8. (Original) The method of claim 1, further comprising the step of:

marking a particular directory as being in the process of conversion once the particular directory is in the list.

9. (Original) The method of claim 8, wherein a new object added to the particular directory is appended at an end of the particular directory.

10. (Previously Presented) The method of claim 1, wherein the step of converting further includes the steps of:

a) creating a second-type root directory;

b) creating a second-type directory corresponding to a particular directory in the list;

c) generating a respective link in the second-type directory for each child object of the particular directory;

d) activating the second-type directory; and

e) removing the particular directory from the list.

11. (Original) The method of claim 10, further comprising the step of:

creating a data structure associated with the second-type directory, the data structure including a first anchor point that is associated with a parent directory of the directory and a second anchor point associated with a parent directory of the second-type directory.

12. (Original) The method of claim 1, further comprising the steps of:

detecting a condition for pausing the converting step; and

in response to the condition, pausing the converting step.

13. (Original) The method of claim 12, wherein the condition is one of:

a product install on the filesystem; a restore operation involving the filesystem; and a back-up operation involving the filesystem.

14. (Original) The method of claim 1, further comprising the steps of:

determining a usage rate of a particular directory before converting that directory; and

postponing converting the particular directory based on the usage rate.

15. (Original) The method of claim 1, further comprising the steps of:

detecting a condition for stopping the converting step; and

in response to the condition, stopping the converting step.

16. (Original) The method of claim 15, wherein the condition is one of:

a system crash, encountering a corrupted object within the filesystem, and insufficient available storage.

17. (Original) The method of claim 15, wherein the converting step is restarted upon a subsequent initial program load involving the filesystem.

18. (Original) The method of claim 3, wherein the step of activating further includes the steps of:

identifying a data structure associated with the directory;

changing the data structure to be associated with the second-type directory; and

removing the directory.

19. (Original) The method of claim 18, further comprising the step of:

asserting a lock on first data structure while performing the step of changing.

20. (Original) A method for converting a filesystem from a first type to a second type, the method comprising the steps of:

executing a conversion process to convert each directory of the first type in the filesystem into the second type while maintaining the filesystem in an active state; and

terminating the conversion process when every directory of the first type in the filesystem has been converted to the second type.

21. (Original) The method of claim 20, further comprising the step of:

initiating the executing step upon initial program load of a computer system utilizing the filesystem.

22. (Original) The method of claim 20, further comprising the step of:

generating a list of directories of the first type in the filesystem to convert.

23. (Original) The method of claim 22, wherein converting each directory includes the steps of:

retrieving an identifier of a directory in the list;

converting the directory to a second-type directory; and

activating the second-type directory.

24. (Original) A program product, comprising:

a program code configured upon execution to:

generate a list of directories of a first type in a filesystem to convert; and convert each directory in the list to a second type while maintaining the filesystem in an active state; and

a signal bearing medium bearing the program code.

25. (Original) A program product, comprising:

a program code configured upon execution to:

initiate a conversion process to convert each directory of a first type in a filesystem into a second type while maintaining the filesystem in an active state; and

terminate the conversion process when every directory of the first type in the filesystem has been converted to the second type; and

a signal bearing medium bearing the program code.

26. (Original) An apparatus comprising:

at least one processor;

a memory coupled with the at least one processor; and

a program code residing in memory and executed by the at least one processor, the program code configured to:

initiate a conversion process to convert each directory of a first type in a filesystem into a second type while maintaining the filesystem in an active state; and

terminate the conversion process when every directory of the first type in the filesystem has been converted to the second type.

27. (Original) The apparatus of claim 26, wherein the program code is further configured to:

non-recursively build a list of directories of the first type.

28. (Original) The apparatus of claim 26, wherein the program code is further configured to:

modify a particular directory being converted such that any new entries in that particular directory are appended at an end of that particular directory.

29. (Original) The apparatus of claim 26, wherein the program code is further configured to:

pause the conversion process if a predetermined operation is detected as being performed on the filesystem.

30. (Original) The apparatus of claim 26, wherein the program code is further configured to:

for a particular directory being converted, maintain a first version having the first type and a second version having the second type; and

activate the second version while de-activating the first version.

31. (Original) The apparatus of claim 26, wherein the program code is further configured to:

stop the conversion process at a particular point when an abnormal system condition is detected; and

restart the conversion process at the particular point.

32. (Original) The apparatus of claim 26, wherein the program code is further configured to:

for a particular directory being converted, determine if the particular directory is being used more than a predetermined amount; and

postpone converting that particular directory while it is being used more than the predetermined amount.

33. (Original) The apparatus of claim 32, wherein the program code is further configured to:

convert another directory, different than the particular directory, while the particular directory is being used more than the predetermined amount.

34. (Original) The apparatus of claim 26, wherein the program code is further configured to:

for a particular directory already converted, convert the particular directory back to the first type.

35. (Original) The apparatus of claim 26, wherein the program code is further configured to:

execute at an adjustable priority level.

IX. EVIDENCE APPENDIX

10/777,869

None.

X. RELATED PROCEEDINGS APPENDIX

10/777,869

None.